Predicting Crimes in Chicago

Shashank Shekhar
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
sshekha4@ncsu.edu

Harshit Badiyani
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
hbbadiya@ncsu.edu

Satanik Ray
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
sray7@ncsu.edu

Nimisha Tripathi
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
ntripat@ncsu.edu

1 Introduction and Background

1.1 Problem Statement

In the past, several studies have been conducted to predict the crime rate in the city of Chicago. These studies have used temporal data to predict the crime rate but they have faced difficulty in discovering the nonlinear relationships and dependencies between data. The main reason for not being able to discover proper relationship in crime rates is the lack of inclusion of the necessary factors governing the crimes.

Present work and motivation involves demystifying the role that social-economic factors play in crime prediction. Crime rates were thought to have only dependent on temporal factors. Here we work on combining Social-Economic Factors with temporal and spatial data to improve on the crimes prediction.

1.2 Related Work

Our work relates to the previous work "Prediction of crime occurrence from multimodal data using deep learning" by Hyeon-Woo Kang and Hang-Bong Kang [1] where they have proposed an accurate crime prediction method by fusing multi-modal data from multiple domains with environmental context information to accurately predict crime occurrences. We have taken ideas from their work where they collect the data from public datasets, transform it and then use Deep Neural Network to extract the Spatial, Temporal and Environmental context feature groups. We modified the approach to use Social-Economic factors as features along with spatial and temporal data to predict the crime rate in the city of Chicago. Instead of using environmental context information in order to make crime predictions, we implemented a Multilayer Perceptron (MLP) based Deep Neural Network Model. For tuning the hyperparameters of this model, we referenced "Understanding deep learning required rethinking generalization" by Chiyuan Zhang, Samy Bengio, Moritz Hardt Bejamin Recht and Oriol Vinyals [2]. In this paper, they have discussed about the maximum number of parameters allowed for a model given the number of hidden layers, neurons per layer and the dimensions of the input data. This paper helped us in ensuring that we can build a model that does not cause data overfitting. For a two-layer dense neural network, the paper proposes that the maximum number of parameters allowed for a model before it starts to overfit is given by ($2*n + d$) where n = number of samples and d is the number of dimensions. With the help of this equation, we could ensure that our model has enough samples and lesser number of parameters to train and hence the model fits the data well. Based on the hyperparameter experiments, we also found out that we didn't need any dropout layer since the model fit the data well in the first place.

Github Link to the Code

2 Method

2.1 Approach

We have used Multilayer Perceptron, Random Forest Regressor and Support Vector Regressor as the models for crime prediction. We will do a comparative analysis to check which model gives better results on the validation set and use that model for making crime predictions on the test data. The reasons explaining why used the above-mentioned specific models are discussed in the Rationale section below.

In the past, attempts to solve the problem of crime prediction have been made by using only the temporal factors. In our approach, we have added social-economic and spatial factors to the temporal factors to be able to make better crime predictions. In our approach, we attempt to consider all the important factors as inputs that govern crime predictions.

2.2 Rationale

We have chosen Multilayer Perceptron as the primary model for this dataset because of the following reasons:

- Large size of the dataset (approx. 2 million records): Neural Networks have the capability to model complex non-linear relationships on large datasets due to their increased flexibility given the higher number of parameters they have.
- Check for Overfitting; This model keeps a check on data overfitting by using dropout as the regularization technique. In our implementation, we have used dropouts for regularization.
- No restriction on the relationships of the input variables: Compared to other techniques such as linear regression which imposes a strict restriction that the inputs must not show any signs of multicollinearity, this model does not impose any such restrictions related to the input data. It can better model data having heteroskedasticity and multicollinearity.

On similar lines, we decided to use Random Forest due to the below reasons:

- It gives more stable and accurate predictions by creating multiple decision trees and merging them together.
- Since it uses multiple decision trees and averaging, it helps to reduce the problem of overfitting and reduces variance.

We have also chosen Support Vector Regressors due to the following reasons:

- High dimensional kernel induced feature space can be used to model the non-linear relationships between the data.
- It is robust to the problem of overfitting.

3 Experiment

3.1 Datasets

Two datasets are used for this project and both are taken from Kaggle.

- Chicago Dataset for Social-Economic Indicators [Period: 2008 to 2012] – This dataset is extracted from the records maintained by the Chicago Department of Health. It consists of 77 records and 9 features of which 7 are the social economic indicators for the city of Chicago. The other two features represent the Community Area Number and the Community Area Name of the corresponding communities for which the social economic indicators are recorded.

- Chicago Crime Dataset [Period: 2001 to 2018] – This dataset is extracted from the records maintained by the Chicago Police Department's Bureau of records. It consists of 3024380 records (approx. 3 million) and 23 features. There are only a few relevant attributes in this dataset that are needed for the purpose of regression analysis for crime prediction and hence these have been merged with social-economic indicators for the purpose of analysis. These attributes include Primary Type, Community Area and an attribute Month that is the transformation of the Date attribute representing the month in which the crime was committed. Other attributes from the original dataset which include Description and Location Description of the crime are used for the purpose of visualizing the time based trends of different types of crimes and a view of the crimes based on the location, type and time.

3.2 Hypotheses

The following hypothesis are made:

- Crimes in the city of Chicago do not simply follow a time-based trend. This means that time alone cannot be taken as a factor to accurately predict the number of crimes in the city. On studying the trends in the crime, it is observed that it is dependent on several social-economic factors. In order to confirm this hypothesis, a pair-plot is made showing trends between crimes and different social-economic indicators. Each of those plots shows a significant positive or negative trend thus denying the direct time dependency of crimes. Hence, a time series model for crimes prediction won't provide accurate predictions, thus requiring the need of other social-economic and spatial factors.
- We hypothesize that the Multilayer Perceptron Model will perform better than the other models due the flexibility of tuning the high number of parameters and its ability to effectively handle data overfitting.

3.3 Experimental Design

In order to predict crimes in the city of Chicago based on a set of Social-Economic and Spatial Indicators, we have merged the data from two different datasets:

- Chicago Dataset for Social-Economic Indicators [Period: Year 2008 to 2012]
- Chicago Crime Dataset [Period: Year 2008 to 2012]

The approach involves two distinct steps:

- Data Preprocessing
- Model Fitting and Hyperparameter tuning

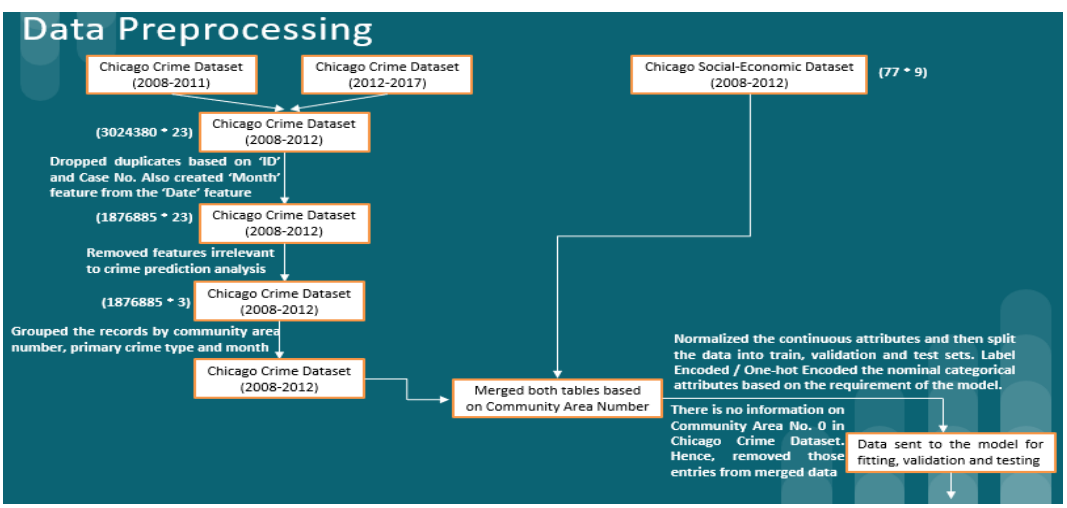

Fig 1: Data Preprocessing Steps

Github Link to the Code

- Data preprocessing - This involves merging the Chicago Crime Dataset (2008 – 2011) with the Chicago Crime Dataset (2012 – 2017) to obtain the Crime records for the period 2008 to 2012. Duplicate records based on ID and Case Number were dropped and Month feature was engineered from the original Date features. Columns not relevant to crime prediction were dropped. These included features like Arrest (Yes/ No) or Domestic Crime (Yes / No). These columns do not influence the crime rate and hence play no role in crime prediction. Thus, they are dropped. Next, we group our records based on Community Area Number, Crime Type and Month in which crimes were committed. Then we merge this data with another dataset containing Social-Economic Indicators for the city of Chicago. After merging, we look for any missing values and find out that the Crime records data does not have any information related to Community Area Number 0. In order to deal with this problem, we drop the corresponding entries related to Community Area number 0 from the merged data. The only drawback we have here is that we cannot make predictions related to community number 0 which is fine if no information about other communities is lost. We then normalize all the continuous attributes and then divide the given dataset into training (80%), validation (10%) and test (10%) sets. At this point, the training data is provided to train the regressor models, validation data to tune the hyperparameters and the test data to make predictions using the best model obtained after hyperparameter tuning.

- For the Model Fitting step, we have used three different regressor models for predicting the crime rates. These are Multilayer Perceptron, Random Forest and Support Vector Regressor Models.

  First model is a Multilayer Perceptron model. For this model, the categorical variables are one-hot encoded (This is done because all the three categorical variables are nominal and thus it is important to make sure that when these are passed as input, they are all treated equally by the activation functions). We then tune the hyperparameters which include the number of hidden layers chosen, the number of neurons per hidden layer and dropout chosen if any. We have used 'Relu' as the activation function for all the hidden layers except the output layer. Since this is a regression problem and the output should predict the crime rate, we have used 'Linear' activation function for the output layer.

  Second model is the Random Forest Regressor Model. For this model, the categorical variables are label encoded. The hyperparameters chosen for this model are the number of estimators and the maximum tree depth.

  Third model is the Support Vector Regressor Model. For this model, the categorical variables are one-hot encoded. The hyperparameters chosen for this model are the values of C, gamma and kernel function used.

  Below tables shows the hyperparameter tuning steps for each of the three models. Mean Squared Error is the metric used for comparative analysis.

  Below figure represents the Mean Squared Error values obtained for the various Hyperparameters chosen for the Multilayer Perceptron Model:

Batch Size: 32

Neurons/ Dropout Rate	0.3	0.5	0.8
32	81.5677	122.209	431.065
64	60.3466	93.916	191.587
128	55.285	59.3568	118.251

Batch Size: 64

Neurons/Dropout Rate	0.3	0.5	0.8
32	124.43	133.675	477.695
64	55.3656	72.1274	219.454
128	54.112	58.3809	100.683

Batch Size: 128

Best Model Hyperparameters

Github Link to the Code

Neurons/ Dropout Rate	0.3	0.5	0.8
32	110.418	148.159	496.428
64	60.373	106.565	207.229
128	52.5242	65.4541	120.79

No. of Layers	No. of Neurons/ Layer	Batch Size	Dropout Rate	Activation Function
2	256	256	0.0	Relu

Fig 2: Hyperparameters for the Multilayer Perceptron Model

We used grid search to tune the hyperparameters for an MLP. On performing the search, it was observed that the model performs better with higher batch size, higher number of neurons per layer and Relu activation Function.

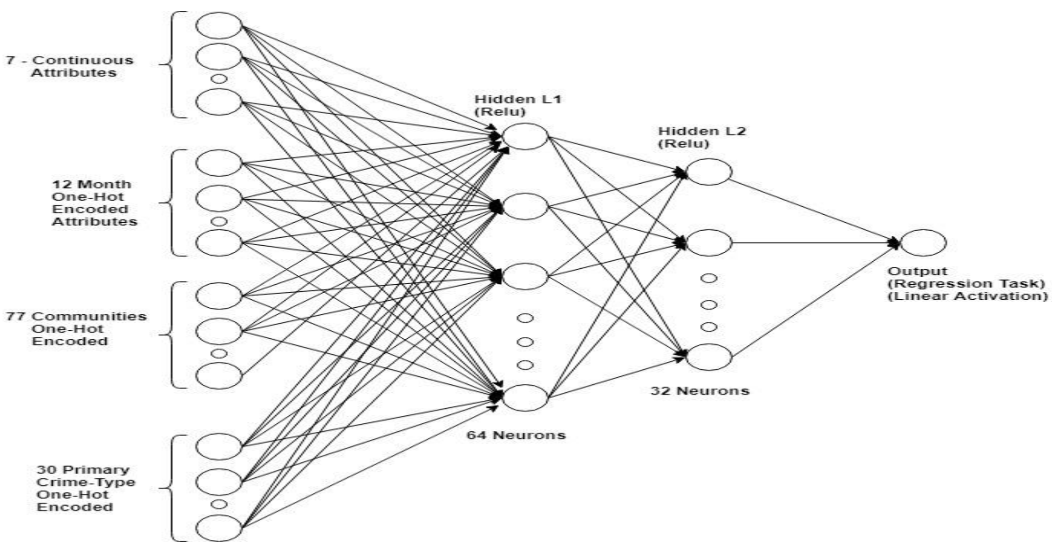

Fig 3: Multilayer Perceptron Architecture

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 64)                8128
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 10,241
Trainable params: 10,241
Non-trainable params: 0
```

Fig 4: Shape of each layer used in MLP

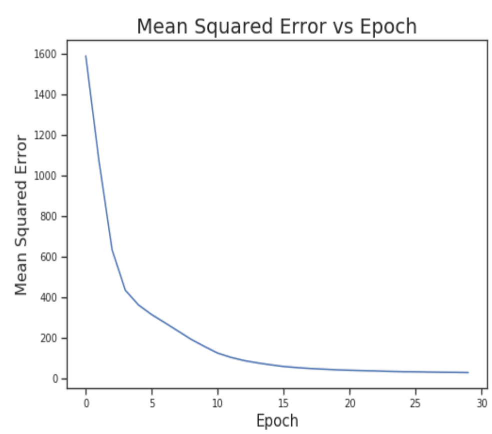

Fig 5: Mean Squared Error vs Epoch

Above graph shows the Mead Squared Error vs Epoch curve. It suggests that the model gets trained at around 15 epochs and the mean squared error has achieved its minimum value. We found that the best model obtained on the validation dataset was with 2 hidden layers, 256 neurons / layer, batch size = 256, Relu activation function and no dropout and it gave a minimum mean squared error of 50.47

[Github Link to the Code]

Fig 6 represents the Mean Squared Error values obtained for the various Hyperparameters chosen for the Random Forest Model:

n_estimators / max_depth	10	20	30	40
10	130.231	67.3674	64.0749	70.5069
25	120.309	69.414	66.7503	64.4874
50	115.966	61.7925	65.4444	63.1005
100	118.751	64.8371	63.0328	63.609

Fig 6: Hyperparameters for the Random Forest Model

Kernel Function = RBF

C / Gamma	0.1	0.001	0.001
0.1	1812.37	1961.95	1988.29
1	1365.06	1730.13	1960.42
10	1065.45	1307.83	1719.95
100	598.666	1140.6	1303.68

Fig 7(a): Hyperparameters for SVR

Kernel Function: Linear, Gamma: auto
Kernel Function: Polynomial, Gamma: scale

Kernel Fn/ C	0.1	1	10	100
Linear	1404.02	1182.40	1148.34	1143.39
Poly	1992.28	1992.13	1990.60	1977.87

Fig 7(b): Hyperparameters for SVR

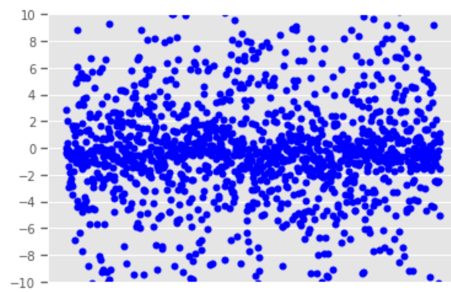

Fig 8: Scatter Plot of the residuals

The best model was obtained for 20 estimators with maximum tree depth of 50 which gave a mean squared error of 61.79

Figures 7(a) and 7(b) represent the Mean Squared Error values obtained for the various Hyperparameters chosen for the Support Vector Regressor Model

From the above table it is observed that the best support vector regressor model for Kernel Function = RBF, C = 100 and Gamma = 0.1 does not perform very well. For our dataset, SVR models do not give good results.

4 Results

4.1 Results

The results obtained for the three different trained models are:

- Multilayer Perceptron: The best model obtained after hyperparameter tuning of the MLP has the following parameters:

    a. 2 hidden layers with Relu activation functions
    b. 256 neurons / layer
    c. Batch Size = 256

[Github Link to the Code]

For this model, the following results were obtained:

Metric Used	Result
MSE	51.91
RMSE	7.20
R2 Prediction Score (Coefficient of Determination)	0.9696
Adjusted R2 Prediction Score	0.9567

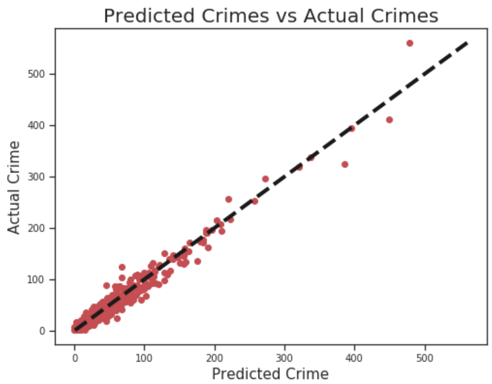

Fig 9: Regression Output on test data for MLP

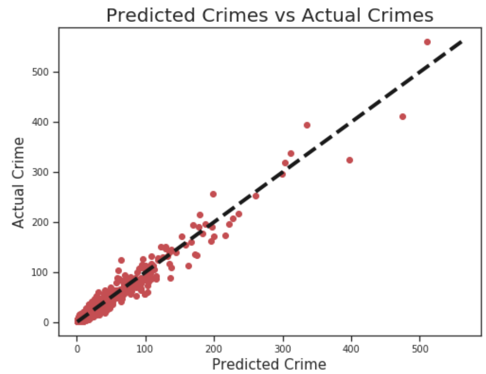

Fig10: Regression Output on test data for Random Forest Model

- Random Forest: The best model obtained after hyperparameter tuning of the Random Forest Regressor has the following parameters:

  a. Number of estimators = 50
  b. Maximum tree depth = 20

  For this model, the following results were obtained:

Metric Used	Result
MSE	63.62
RMSE	7.97
R2 Prediction Score (Coefficient of Determination)	0.9610
Adjusted R2 Prediction Score	0.9574

- Support Vector Regressor: The best model obtained after hyperparameter tuning for the Support Vector Regressor has the following parameters:

  a. Kernel Function: RBF
  b. C = 100
  c. Gamma = 0.1

[Github Link to the Code]

For this model, the following results were obtained:

Metric Used	Result
MSE	598.67
RMSE	24.47
R2 Prediction Score (Coefficient of Determination)	-0.35
Adjusted R2 prediction score	-0.47

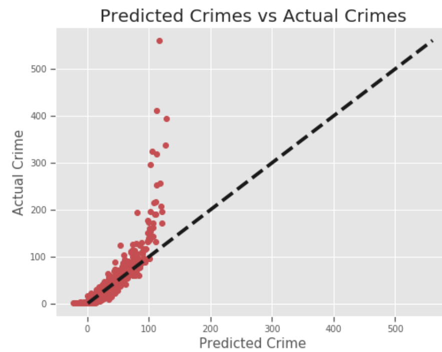

Fig 11: SVR Metrics table and Regression Output

- MLP and Random Forest have high R2 and Adjusted R2 score and had a random distribution of the residuals indicating a good fit of data. This means that both models can be used to make accurate crime predictions given a set of Social-Economic indicators. Support Vector Regressor model had a negative R2 score indicating that the model performs worse than predicting the average. Hence, it is dropped from further analysis and not used for crime predictions.

4.2 Discussion

The results of the experiment can accurately predict crimes given a set of Social-Economic Indicators. The predictions on the test dataset for both the MLP and Random Forest Models are shown in figures 9 and 10 respectively. Improvements in the result can be made by using the principles of Model Stacking which can take the best predictions of both the models. Currently, we faced a limitation to implement this due to the compatibility issues of the keras based MLP model with the scikit-learn based Random Forest model.

We also find that both the hypothesis that we initially made holds true:

- Pair-plot shown in Fig 12 (Appendix) confirms that the crimes in the city don't just depend on temporal factors but also depend on the social-economic factors.
- Multilayer Perceptron model gives the best results.

4.3 Future Work and Improvements

- Some features from the original data like latitude and longitude can be used together with maps to provide the public with the safest travel routes during specific time of day.
- The same information can be shared with the Police Department which can help in proper deployment of cops at different places within the city at specific times of day.

5 Conclusion

This project helped us understand the relationship between the problem given and the model used to solve the problem. Different models are suitable to different types of problems and it requires a deep analysis of the problem at hand to understand which model should be used to solve a given problem. We also learned from this project that there are some important considerations while building the model. Knowing the limitations of the model used and adopting methods to deal with them can help in getting better results.

Github Link to the Code

References

[1] Hyeon-Woo Kang, Hanh-Bong Kang (2017), Prediction of crime occurrence from multi-modal data using deep learning.

[2] Understanding deep learning required rethinking generalization by Chiyuan Zhang, Samy Bengio, Moritz Hardt Bejamin Recht and Oriol Vinyals.

[3] Crime Rate Inference with Big DataCrime Rate Inference with Big Data by Hongjian Wang, Daniel Kifer, Corina Graif, Zhenhui Li

Appendix

Fig 12: Relationship between Social-Economic Indicators and Crimes

[Github Link to the Code]